Unit wise FAQs

Sub- Microprocessor and Microcontroller

UNIT 1: Fundamentals of 8085 Microprocessor

1. Explain the architecture of the 8085 microprocessors with a neat functional description.

The **8085 microprocessor** is an 8-bit microprocessor developed by Intel. It is capable of addressing up to 64 KB of memory using a 16-bit address bus. The architecture of 8085 consists of several functional blocks that work together to execute instructions.

At the core of the processor is the **Arithmetic Logic Unit (ALU)**. The ALU performs arithmetic operations such as addition, subtraction, increment, and decrement, and logical operations such as AND, OR, XOR, compare, and complement. The ALU works closely with the **Accumulator (A register)**, which stores intermediate results of operations.

The 8085 contains six general-purpose registers: B, C, D, E, H, and L. These registers are 8-bit registers but can be combined as register pairs (BC, DE, HL) for 16-bit operations. The **Program Counter (PC)** is a 16-bit register that holds the address of the next instruction to be executed. The **Stack Pointer (SP)** is another 16-bit register that points to the top of the stack in memory.

The **Instruction Register and Decoder** fetch the opcode from memory and decode it to generate necessary control signals. The **Timing and Control Unit** generates timing signals required for internal and external operations.

The processor communicates with memory and I/O devices through three buses:

- **Address Bus (16-bit)**
- **Data Bus (8-bit)**
- **Control Bus**

Interrupts such as TRAP, RST 7.5, RST 6.5, RST 5.5, and INTR allow external devices to request processor service. The serial I/O control and clock generator are also part of the architecture.

Overall, the 8085 architecture is simple and efficient, forming the basis of early embedded and control systems.

2. Explain the bus organization of 8085 microprocessor.

The 8085 microprocessor uses three major buses to communicate with memory and peripheral devices: Address Bus, Data Bus, and Control Bus.

The **Address Bus** is 16-bit wide and unidirectional. It carries the memory address from the microprocessor to memory or I/O devices. Since it is 16-bit wide, the processor can access 2^{16} = 65,536 memory locations (64 KB).

The lower 8 bits of the address bus (AD0–AD7) are multiplexed with the data bus to reduce the number of pins. The **Address Latch Enable (ALE)** signal is used to demultiplex these lines.

The **Data Bus** is 8-bit wide and bidirectional. It carries data between the processor and memory or I/O devices. The bidirectional nature allows the processor to both read and write data.

The **Control Bus** consists of various control signals such as RD (Read), WR (Write), IO/M (selects memory or I/O operation), READY (wait state control), HOLD and HLDA (DMA control).

This bus organization ensures proper synchronization and efficient data transfer between CPU, memory, and peripherals.

3. Explain the register organization of 8085 including flag register.

The 8085 contains several registers categorized as general-purpose and special-purpose registers. The general-purpose registers include B, C, D, E, H, and L. These are 8-bit registers used to temporarily store data during program execution. They can also be paired as BC, DE, and HL to perform 16-bit operations. The HL pair often acts as a memory pointer.

The special-purpose registers include the Accumulator (A), Program Counter (PC), Stack Pointer (SP), Instruction Register, and Flag Register.

The **Flag Register** is an 8-bit register containing five active flags:

- **Sign (S)**: Set if result is negative
- **Zero (Z)**: Set if result is zero
- **Auxiliary Carry (AC)**: Used in BCD operations
- **Parity (P)**: Set if result has even parity

- **Carry (CY)**: Set if carry is generated

These flags are automatically modified after arithmetic and logical operations. They help in decision-making instructions like conditional jumps.

The Program Counter holds the address of the next instruction, while the Stack Pointer points to stack memory used during subroutines and interrupts.

4. Explain addressing modes of 8085 with examples.

Addressing modes specify how the operand of an instruction is accessed. The 8085 supports five main addressing modes.

1. **Immediate Addressing Mode**
   The operand is directly specified in the instruction.
   Example: MVI A, 45H
2. **Register Addressing Mode**
   The operand is stored in a register.
   Example: MOV A, B
3. **Direct Addressing Mode**
   The memory address is directly specified.
   Example: LDA 2500H
4. **Register Indirect Addressing Mode**
   The memory address is specified indirectly by register pair.
   Example: MOV A, M (HL pair holds address)
5. **Implied Addressing Mode**
   Operand is implied.
   Example: CMA

These addressing modes provide flexibility in accessing data.

5. Explain instruction set classification of 8085.

The instruction set of 8085 is classified into five categories:

1. **Data Transfer Instructions** (MOV, MVI, LDA, STA)
2. **Arithmetic Instructions** (ADD, SUB, INR, DCR)

3. **Logical Instructions** (ANA, ORA, XRA, CMP)
4. **Branching Instructions** (JMP, JZ, JNZ, CALL, RET)
5. **Machine Control Instructions** (HLT, NOP, EI, DI)

Each category performs specific tasks. Data transfer moves data, arithmetic performs calculations, logical handles bitwise operations, branching controls program flow, and machine control manages processor states.

6. Explain stack and subroutine in 8085.

The stack is a memory area organized in LIFO (Last In First Out) manner. It is used to store return addresses, register contents, and temporary data. The Stack Pointer (SP) points to the top of stack.

During a subroutine call (CALL instruction), the current Program Counter content is pushed onto stack. Control transfers to subroutine address. After execution, RET instruction pops the return address back into PC.

Stack operations include PUSH and POP instructions. Stack grows downward in memory.

Subroutines improve modularity and code reusability.

7. Explain interrupt structure of 8085.

Interrupts are signals that temporarily suspend the main program to service external events. The 8085 has five interrupts:

- TRAP (highest priority, non-maskable)
- RST 7.5
- RST 6.5
- RST 5.5
- INTR (lowest priority)

RST 7.5, 6.5, and 5.5 are maskable vectored interrupts. TRAP cannot be disabled. INTR is non-vectored.

Interrupt Enable (EI) and Disable (DI) instructions control maskable interrupts.

Priority order: TRAP > RST 7.5 > RST 6.5 > RST 5.5 > INTR.

8. Explain assembly language programming structure in 8085.

An assembly program consists of:

- **Label**
- **Opcode**
- **Operand**
- **Comment**

Example:

START: MVI A, 05H
ADD B
HLT

Assembler directives like ORG, DB, END are used. Programs follow sequence: load data, process, store result, stop.

9. Explain machine cycles and instruction cycles in 8085.

Instruction cycle is time taken to complete one instruction. It consists of machine cycles such as:

- Opcode Fetch
- Memory Read
- Memory Write
- I/O Read
- I/O Write

Each machine cycle consists of T-states. Opcode fetch requires 4–6 T-states.

Timing diagram explains synchronization.

SHORT QUESTIONS (10)

1. What is word length of 8085?
   → 8-bit
2. Maximum memory?
   → 64 KB
3. Size of address bus?
   → 16-bit
4. Name flag bits.
   → S, Z, AC, P, CY
5. Highest priority interrupt?
   → TRAP
6. Function of ALE?
   → Demultiplex address/data bus
7. Stack works on which principle?
   → LIFO
8. Size of data bus?
   → 8-bit
9. What is opcode?
   → Operation part of instruction
10. What is NOP?
    → No operation instruction

UNIT 2: Advanced Microprocessor – 8086

1. Explain the architecture of the 8086 microprocessors in detail.

The **8086 microprocessor** is a 16-bit processor developed by Intel. It has a 16-bit data bus and a 20-bit address bus, allowing it to access up to 1 MB (2^{20} bytes) of memory. The architecture of 8086 is divided into two major functional units: the **Execution Unit (EU)** and the **Bus Interface Unit (BIU)**. This division enables pipelined processing, improving system performance.

The **Execution Unit (EU)** is responsible for decoding and executing instructions. It contains the Arithmetic Logic Unit (ALU), general-purpose registers (AX, BX, CX, DX), pointer and index registers (SP, BP, SI, DI), and the Flag Register. The EU takes instructions from the instruction queue, decodes them, and performs arithmetic, logical, or control operations. The ALU performs operations such as addition, subtraction, multiplication, division, AND, OR, XOR, and compare.

The **Bus Interface Unit (BIU)** handles all bus operations such as instruction fetching, memory reading/writing, and I/O operations. It contains the segment registers (CS, DS, SS, ES), Instruction Pointer (IP), and a 6-byte instruction queue. The BIU calculates the physical address using the formula:

Physical Address = Segment × 10H + Offset

This segmentation allows efficient memory management.

The instruction queue enables **pipelining**, where BIU fetches instructions while EU executes them simultaneously, thus increasing throughput.

The architecture supports minimum and maximum mode operations for single and multiprocessor systems.

Overall, the 8086 architectures marked a major advancement over 8085 by introducing segmentation and pipelining, forming the foundation of modern x86 processors.

2. Explain memory segmentation and physical address calculation in 8086.

The 8086 microprocessor uses a segmented memory architecture to access 1 MB of memory using 16-bit registers. Since a 16-bit register can address only 64 KB, segmentation is used to overcome this limitation.

Memory is divided into logical segments of 64 KB each. There are four main segment registers:

- Code Segment (CS) – stores program instructions
- Data Segment (DS) – stores data
- Stack Segment (SS) – stores stack data
- Extra Segment (ES) – additional data storage

Each segment register holds a 16-bit base address. The actual physical address is calculated by shifting the segment address left by 4 bits (multiplying by 10H) and adding a 16-bit offset.

Physical Address = Segment × 10H + Offset

For example, if CS = 2000H and IP = 3456H:

Physical Address = 2000H × 10H + 3456H
= 20000H + 3456H
= 23456H

This scheme allows overlapping segments and flexible memory management.

Segmentation provides modular programming, efficient memory usage, and logical separation of code, data, and stack.

3. Describe the register organization of 8086.

The 8086 has fourteen 16-bit registers grouped as:

1. General Purpose Registers:

- AX (Accumulator)
- BX (Base Register)
- CX (Count Register)
- DX (Data Register)

Each can be split into high and low bytes (AH/AL, BH/BL, etc.).

2. Pointer and Index Registers:

- SP (Stack Pointer)
- BP (Base Pointer)
- SI (Source Index)
- DI (Destination Index)

These are used for memory addressing and string operations.

3. Segment Registers:

- CS (Code Segment)
- DS (Data Segment)
- SS (Stack Segment)
- ES (Extra Segment)

4. Instruction Pointer (IP):

Holds offset address of next instruction.

5. Flag Register:

Contains status and control flags:

- Carry (CF)
- Parity (PF)
- Auxiliary Carry (AF)
- Zero (ZF)
- Sign (SF)
- Overflow (OF)
- Direction (DF)
- Interrupt (IF)

- Trap (TF)

This register organization supports powerful instruction execution and memory addressing.

4. Explain minimum and maximum mode operations of 8086.

The 8086 operates in two modes:

Minimum Mode:

Used for single-processor systems. The processor generates all control signals internally. Pins such as RD, WR, M/IO, INTA are directly available. It simplifies hardware design.

Maximum Mode:

Used for multiprocessor systems. An external bus controller (8288) generates control signals. Signals like S0, S1, S2 indicate bus status. It supports coprocessor (8087).

Maximum mode enables advanced configurations and higher performance systems.

5. Explain the bus cycle of 8086.

A bus cycle consists of four T-states (T1–T4). During:

- T1: Address placed on bus
- T2: Read/Write signal activated
- T3: Data transfer
- T4: Completion of operation

Wait states may be inserted if memory is slow.

Bus cycles include:

- Memory Read
- Memory Write
- I/O Read
- I/O Write
- Interrupt Acknowledge

6. Explain addressing modes of 8086 with examples.

8086 supports various addressing modes:

- Immediate (MOV AX, 1234H)
- Register (MOV AX, BX)
- Direct (MOV AX, [2000H])
- Register Indirect (MOV AX, [BX])
- Based
- Indexed
- Based Indexed
- Relative

These provide flexible data access.

7. Discuss instruction set of 8086.

Instruction set categories:

- Data Transfer (MOV, PUSH, POP)
- Arithmetic (ADD, SUB, MUL, DIV)
- Logical (AND, OR, XOR, NOT)
- String (MOVS, CMPS)
- Control Transfer (JMP, CALL, RET)
- Processor Control (HLT, NOP)

Supports signed and unsigned operations.

8. Compare 8085 and 8086.

Feature	**8085**	**8086**
Data Bus	8-bit	16-bit

Feature	8085	8086
Address Bus	16-bit	20-bit
Memory	64 KB	1 MB
Pipelining	No	Yes
Registers	8-bit	16-bit

8086 is more powerful and faster.

SHORT QUESTIONS (10)

1. Data bus size of 8086?
   → 16-bit
2. Address bus size?
   → 20-bit
3. Maximum memory access?
   → 1 MB
4. Number of segment registers?
   → Four
5. Physical address formula?
   → Segment × 10H + Offset
6. Queue size?
   → 6 bytes
7. Modes of operation?
   → Minimum & Maximum
8. Flag register size?
   → 16-bit
9. Hardware interrupts?
   → INTR, NMI
10. What enables pipelining?
    → Instruction queue

UNIT 3: Introduction to 8051 Microcontroller

1. Explain the architecture of the 8051 microcontrollers in detail.

The **8051 microcontroller** is an 8-bit microcontroller designed by Intel, widely used in embedded systems. Unlike a microprocessor, which requires external memory and peripherals, the 8051 integrates CPU, RAM, ROM, timers, serial port, and I/O ports on a single chip, making it suitable for real-time control applications.

The architecture of the 8051 consists of the following major blocks:

1. Central Processing Unit (CPU)

The CPU includes the Arithmetic Logic Unit (ALU), Accumulator (A), B register, Program Status Word (PSW), Stack Pointer (SP), and Program Counter (PC). The ALU performs arithmetic and logical operations. The accumulator is the primary register for arithmetic operations, while the B register is mainly used during multiplication and division.

2. Memory Organization

The 8051 follows Harvard architecture, meaning program memory and data memory are separate.

- Program Memory: Typically, 4 KB on-chip ROM (expandable to 64 KB externally).
- Data Memory: 128 bytes of internal RAM, expandable to 64 KB external RAM.

The internal RAM is divided into register banks, bit-addressable area, and general-purpose RAM.

3. I/O Ports

The 8051 has four 8-bit I/O ports (P0–P3), allowing direct interfacing with external devices. Some pins have dual functions such as serial communication, interrupt inputs, and timer inputs.

4. Timers and Counters

It includes two 16-bit timers/counters (Timer 0 and Timer 1) used for delay generation, event counting, and baud rate generation.

5. Serial Communication

A full-duplex UART enables serial communication using TXD and RXD pins.

6. Interrupt System

The 8051 supports five interrupt sources with two priority levels.

Overall, the 8051 architecture integrates multiple peripherals, reducing hardware complexity and making it ideal for embedded control systems.

2. Explain memory organization of 8051 in detail.

The memory organization of the 8051 is divided into program memory and data memory, following Harvard architecture.

Program Memory

The standard 8051 contains 4 KB of on-chip ROM used to store program instructions. It can access up to 64 KB of external program memory. The Program Counter (PC) is 16-bit, allowing addressing of 64 KB.

Data Memory

The 8051 contains 128 bytes of internal RAM, divided into:

1. Register Banks (00H–1FH):
   There are four register banks (R0–R7), each containing eight registers.
2. Bit-Addressable Area (20H–2FH):
   16 bytes (128 bits) are individually addressable, useful for control applications.
3. General-Purpose RAM (30H–7FH):
   Used for temporary data storage.

Special Function Registers (SFRs)

Located from 80H to FFH, these registers control timers, ports, serial communication, interrupts, etc.

External Data Memory

Up to 64 KB can be accessed using MOVX instruction.

This organized structure allows efficient program execution and peripheral control.

3. Describe Special Function Registers (SFRs) of 8051.

Special Function Registers (SFRs) are control registers located in upper memory space (80H–FFH). They control internal peripherals.

Important SFRs include:

- ACC (Accumulator): Stores arithmetic results.
- B Register: Used in multiplication and division.
- PSW (Program Status Word): Contains flags like CY, AC, OV, P.
- SP (Stack Pointer): Points to stack location.
- DPTR (Data Pointer): 16-bit register for external memory access.
- TMOD and TCON: Control timer modes and operations.
- SCON and SBUF: Control serial communication.
- IE and IP: Control interrupt enable and priority.
- Port Registers (P0–P3): Used for I/O operations.

These SFRs enable control over internal hardware modules.

4. Explain I/O port structure of 8051.

The 8051 has four 8-bit I/O ports. Each port has an internal latch and driver circuit.

- Port 0: Dual-purpose port (address/data bus during external memory access).
- Port 1: General-purpose I/O.
- Port 2: High-order address bus in external memory mode.
- Port 3: Multipurpose port with alternate functions (INT0, INT1, RXD, TXD, T0, T1, WR, RD).

Each pin can be configured as input or output by writing 1 or 0 to port latch.

5. Explain interrupt system of 8051.

The 8051 supports five interrupt sources:

- External Interrupt 0

- Timer 0
- External Interrupt 1
- Timer 1
- Serial Communication

Interrupt Enable (IE) register controls activation. Interrupt Priority (IP) register sets priority level.

When an interrupt occurs:

1. Current instruction completes.
2. PC is pushed to stack.
3. Control jumps to interrupt vector address.
4. ISR executes.
5. RETI returns to main program.

Interrupts improve real-time response.

6. Explain timers and counters of 8051.

The 8051 contains two 16-bit timers/counters: Timer 0 and Timer 1.

They operate in four modes:

- Mode 0: 13-bit timer
- Mode 1: 16-bit timer
- Mode 2: 8-bit auto-reload
- Mode 3: Split timer mode

Timers generate delays, measure time intervals, or count external events.

TMOD configures mode, TCON controls start/stop.

7. Explain serial communication in 8051.

The 8051 supports full-duplex serial communication through UART.

SCON register controls operation modes:

- Mode 0: Shift register
- Mode 1: 8-bit UART
- Mode 2: 9-bit UART
- Mode 3: 9-bit variable baud rate

SBUF register stores data for transmission and reception.

Timer 1 usually generates baud rate.

8. Compare microprocessor and microcontroller.

Microprocessor:

- Only CPU
- Requires external memory and peripherals
- Used in general computing

Microcontroller:

- CPU + Memory + I/O + Timers on chip
- Used in embedded systems
- Lower cost and power

9. Explain pin configuration of 8051.

The 8051 has 40 pins:

- 4 Ports (32 pins)
- VCC and GND
- XTAL1, XTAL2 (clock)
- RST (reset)
- PSEN, ALE, EA (external memory control)

10. Explain advantages and applications of 8051.

Advantages:

- Low cost
- On-chip peripherals
- Easy programming
- Widely supported

Applications:

- Industrial automation
- Traffic control systems
- Home appliances
- Robotics
- Embedded systems

SHORT QUESTIONS (10)

1. Word length of 8051? → 8-bit
2. Internal RAM size? → 128 bytes
3. On-chip ROM? → 4 KB
4. Number of I/O ports? → Four
5. Timers available? → Two
6. Interrupt sources? → Five
7. Architecture type? → Harvard
8. Size of DPTR? → 16-bit
9. Serial communication type? → Full duplex
10. Stack pointer default value? → 07H

UNIT 4: 8051 Assembly Language Programming

1. Explain addressing modes of 8051 with examples.

The 8051 microcontroller supports several addressing modes that determine how operands are accessed during instruction execution.

1. Immediate Addressing Mode

The operand is directly specified in the instruction using the "#" symbol.
Example:
MOV A, #55H
Here, 55H is loaded directly into accumulator.

2. Register Addressing Mode

The operand is located in a register (R0–R7).
Example:
MOV A, R1

3. Direct Addressing Mode

The address of operand is directly specified.
Example:
MOV A, 30H

4. Register Indirect Addressing Mode

The address is stored in R0 or R1.
Example:
MOV A, @R0

5. Indexed Addressing Mode

Used mainly for accessing lookup tables in program memory.
Example:
MOVC A, @A+DPTR

These addressing modes provide flexibility in data access and memory management.

2. Explain data transfer instructions of 8051.

Data transfer instructions move data between registers, memory, and peripherals without altering the original data.

Examples include:

- MOV A, R1
- MOV DPTR, #2000H
- PUSH 30H
- POP 30H
- XCH A, R2

These instructions are essential for initializing registers, transferring data to ports, and managing stack operations.

3. Explain arithmetic instructions of 8051.

Arithmetic instructions perform mathematical operations.

Examples:

- ADD A, R0
- ADDC A, #05H
- SUBB A, R1
- INC A
- DEC R2
- MUL AB
- DIV AB

The PSW flags (CY, AC, OV, P) are affected during these operations. Multiplication stores result in A and B. Division stores quotient in A and remainder in B.

4. Explain logical instructions of 8051.

Logical instructions perform bitwise operations.

Examples:

- ANL A, R1
- ORL A, #0FH
- XRL A, R2
- CLR A
- CPL A
- RL A
- RR A

These are widely used in masking, setting, clearing bits, and data manipulation.

5. Explain branching instructions of 8051 in detail with examples.

Branching instructions control the flow of program execution by altering the Program Counter (PC). They are used for decision-making, looping, subroutine calling, and conditional execution.

Branching instructions can be classified into:

1. Unconditional Branching

These transfer control without checking any condition.
Example:
SJMP LABEL – Short jump within ±128 bytes.
AJMP LABEL – Absolute jump within 2 KB page.
LJMP LABEL – Long jump anywhere in 64 KB memory.

2. Conditional Branching

These depend on flags or bit status.

Examples:

- JZ LABEL – Jump if accumulator is zero

- JNZ LABEL – Jump if not zero
- JC LABEL – Jump if carry flag is set
- JNC LABEL – Jump if carry not set
- JB bit, LABEL – Jump if bit is set
- JNB bit, LABEL – Jump if bit is not set
- DJNZ R0, LABEL – Decrement register and jump if not zero

3. Subroutine Instructions

- CALL – Calls subroutine
- RET – Returns from subroutine

When CALL is executed:

1. Current PC value is pushed onto stack.
2. PC loads subroutine address.
3. After execution, RET pops PC back.

Branching instructions are essential in implementing loops, conditional statements (if-else), and program modularity. For example, implementing a delay loop:

MOV R0, #10

LOOP: DJNZ R0, LOOP

This decrements R0 until zero.

Branching instructions optimize memory usage and enable structured programming in embedded systems.

6. Explain bit manipulation instructions of 8051 in detail.

The 8051 microcontroller has powerful bit-addressable features. It allows manipulation of individual bits in RAM, SFRs, and ports.

Bit-Level Instructions:

1. SETB bit – Sets specified bit

2. CLR bit – Clears specified bit
3. CPL bit – Complements bit
4. JB bit, LABEL – Jump if bit set
5. JNB bit, LABEL – Jump if bit not set
6. JBC bit, LABEL – Jump if bit set and clear it

Example:

SETB P1.0

CLR P1.0

This turns LED ON/OFF connected to port pin.

Bit manipulation is highly useful in embedded systems for controlling devices such as switches, relays, sensors, and communication lines.

Advantages:

- Efficient memory usage
- Faster execution
- Precise control of hardware

Applications include:

- Traffic light control
- LED matrix display
- Stepper motor control

Because of bit-addressable memory area (20H–2FH) and many SFR bits, 8051 becomes ideal for control-based applications.

7. Explain assembly language programming structure in 8051 with example.

Assembly language programming in 8051 consists of instructions written in mnemonic form. The structure includes:

1. Label
2. Opcode
3. Operand
4. Comment

Example program to add two numbers:

```
ORG 0000H
MOV A, #25H
MOV R1, #15H
ADD A, R1
MOV 30H, A
END
```

Explanation:

- ORG sets starting address.
- MOV loads values.
- ADD performs addition.
- Result stored in RAM location 30H.

Steps in assembly programming:

1. Define problem
2. Choose registers
3. Write algorithm
4. Convert to mnemonics
5. Assemble and debug

Stack usage must be managed properly. Interrupt routines must end with RETI.

Assembly language provides full control over hardware and efficient memory utilization, making it suitable for time-critical embedded applications.

8. Explain use of loops and delay generation using 8051 assembly.

Delay generation is crucial in embedded systems for timing control. Delays can be generated using:

1. Software Delay (Loop Method)

Example:

```
MOV R0, #250
L1: MOV R1, #250
L2: DJNZ R1, L2
DJNZ R0, L1
```

Nested loops create larger delays.

Advantages:

- Simple
- No hardware required

Disadvantages:

- CPU busy during delay
- Less accurate

2. Timer-Based Delay

Using Timer 0:

```
MOV TMOD, #01H
MOV TH0, #3CH
MOV TL0, #B0H
SETB TR0
WAIT: JNB TF0, WAIT
CLR TR0
```

CLR TF0

Timer method is accurate and efficient.

Applications:

- LED blinking
- Serial communication baud rate generation
- Motor speed control

Timers are preferred in real-time systems.

9. Explain development of a simple LED interfacing program using 8051 assembly.

Interfacing LEDs with 8051 involves connecting LED to port pins through current limiting resistors.

Example: LED connected to P1.0

Program to blink LED:

```
ORG 0000H

MAIN:

CLR P1.0

ACALL DELAY

SETB P1.0

ACALL DELAY

SJMP MAIN


DELAY:

MOV R0, #200

L1: MOV R1, #255

L2: DJNZ R1, L2

DJNZ R0, L1
```

```
RET

END
```

Explanation:

- CLR P1.0 turns LED ON (active low).
- SETB P1.0 turns LED OFF.
- Delay subroutine controls blinking rate.

This demonstrates:

- Port configuration
- Bit manipulation
- Subroutine call
- Looping

LED interfacing is fundamental in embedded labs and demonstrates practical use of assembly programming.

10. Explain comparison between C programming and Assembly in 8051.

Assembly:

- Fast execution
- Efficient memory usage
- Hardware-specific
- Difficult debugging

C Programming:

- Easy coding
- Portable
- Larger memory usage
- Slower compared to assembly

Embedded systems often use C with inline assembly for optimization.

SHORT QUESTIONS

1. Symbol used for immediate data? → #
2. Which registers support indirect addressing? → R0 and R1
3. Instruction to set bit? → SETB
4. Instruction to clear accumulator? → CLR A
5. Jump if zero? → JZ
6. Decrement and jump? → DJNZ
7. Return from subroutine? → RET
8. Return from interrupt? → RETI
9. Bit-addressable RAM range? → 20H–2FH
10. Timer mode register? → TMOD

UNIT 5: Real World Applications

1. Explain the architecture and operating modes of 8255 Programmable Peripheral Interface.

The **8255 Programmable Peripheral Interface (PPI)** is a widely used peripheral device designed to interface input/output devices with microprocessors such as 8085 and 8086. It provides 24 programmable I/O lines organized into three 8-bit ports: Port A, Port B, and Port C.

The internal architecture of 8255 consists of:

- Data Bus Buffer
- Read/Write Control Logic
- Group A Control
- Group B Control
- Ports A, B, and C

Port A (8-bit) and Port B (8-bit) are independent I/O ports. Port C (8-bit) can be split into two 4-bit ports (PC upper and PC lower). Ports are divided into two groups:

- Group A: Port A + PC upper
- Group B: Port B + PC lower

The 8255 operates in three main modes:

Mode 0 – Basic Input/Output Mode

Ports function as simple input or output ports without handshaking. Suitable for LED, switch interfacing.

Mode 1 – Strobed Input/Output Mode

Supports handshaking signals for synchronized data transfer. Port C lines are used for control signals like STB, IBF, OBF, ACK.

Mode 2 – Bidirectional Mode

Only applicable to Port A. It allows bidirectional data transfer with handshaking.

The Control Word Register configures modes and port directions. Writing a control word determines whether ports operate as input or output.

Applications include keyboard interfacing, display control, ADC/DAC interfacing, and industrial control systems. The 8255 provides flexibility and ease of peripheral interfacing in microprocessor-based systems.

2. Explain the architecture and modes of operation of 8254 Programmable Interval Timer.

The **8254 Programmable Interval Timer (PIT)** is used for precise time delay generation, frequency division, event counting, and waveform generation. It contains three independent 16-bit counters: Counter 0, Counter 1, and Counter 2.

Each counter consists of:

- Count Register
- Control Logic
- Input Clock (CLK)
- Gate Input
- Output Pin

The Control Word Register selects operating mode and counter configuration.

The 8254 supports six modes:

Mode 0 – Interrupt on Terminal Count

Output goes high when count reaches zero.

Mode 1 – Programmable One-Shot

Generates single pulse.

Mode 2 – Rate Generator

Divides input frequency.

Mode 3 – Square Wave Generator

Generates square wave output.

Mode 4 – Software Triggered Strobe

Output pulse after count expires.

Mode 5 – Hardware Triggered Strobe

The counters decrement based on clock pulses. When count reaches zero, output signal changes as per selected mode.

Applications include:

- Baud rate generation
- Real-time clock
- Frequency generation
- Motor speed control
- Traffic light timing

The 8254 improves timing accuracy in embedded systems.

3. Explain the architecture and working of 8257 DMA controller.

The **8257 DMA (Direct Memory Access) Controller** allows data transfer between memory and peripherals without CPU intervention, increasing system efficiency.

The 8257 contains:

- Four DMA Channels
- Data Bus Buffer
- Control Logic
- Address Register
- Count Register
- Priority Resolver

DMA process:

1. Peripheral sends DMA request (DRQ).
2. 8257 sends HOLD request to CPU.

3. CPU responds with HLDA (Hold Acknowledge).
4. DMA takes control of bus.
5. Data transferred directly.
6. DMA releases bus.

Modes:

- Burst Mode
- Cycle Stealing Mode

Advantages:

- Faster data transfer
- Reduces CPU overhead
- Efficient for high-speed devices

Applications:

- Disk data transfer
- ADC/DAC data acquisition
- Video memory transfer

DMA enhances system performance in real-time systems.

4. Explain interfacing of LED with microprocessor/microcontroller.

LED interfacing involves connecting LEDs to output ports through current limiting resistors.

Connection:

- LED anode to port pin via resistor
- Cathode to ground (active high)
  or vice versa (active low)

Program logic:

- Write 1 to port → LED ON
- Write 0 → LED OFF (depending on configuration)

Example:

MOV A, 0FFH

OUT PORT

Applications:

- Status indication
- Binary display
- System diagnostics

LED interfacing is fundamental in embedded labs.

5. Explain interfacing of array of LEDs.

An LED array consists of multiple LEDs connected to different port pins. It can display binary patterns.

Connection:
Each LED connected to one port line with resistor.

Program example:

MOV A, #55H

OUT PORT

Pattern generation using loops creates running LED effect.

Applications:

- Traffic control
- Digital counters
- Decorative lighting

6. Explain interfacing of 7-segment display in detail.

A 7-segment display contains seven LEDs (a–g) arranged to display digits 0–9. It can be common anode or common cathode type.

Each segment is connected to port pins. A lookup table is used to display digits.

Example codes for common cathode:

0 → 3FH

1 → 06H

2 → 5BH

Multiplexing technique allows multiple displays using fewer ports.

Applications:

- Digital clocks
- Calculators
- Frequency counters

Proper current limiting resistors are necessary.

7. Explain interfacing and working of stepper motor.

A stepper motor rotates in discrete steps. It requires sequential excitation of coils.

Interfacing:
Motor driver (ULN2003) used between microcontroller and motor.

Sequence:
Phase A → Phase B → Phase C → Phase D

Example:

MOV A, #09H

OUT PORT

Modes:

- Full Step
- Half Step

Applications:

- Robotics

- CNC machines
- Printers

Stepper motors provide precise position control.

8. Explain switch interfacing.

Switch connected to input port with pull-up resistor.

Logic:
Pressed → 0
Released → 1

Program polls switch and performs action.

Used in keypad systems and control panels.

9. Explain role of programmable peripherals in embedded systems.

Programmable peripherals enhance system capability by providing configurable I/O, timing, and data transfer features.

Benefits:

- Flexibility
- Reduced CPU load
- Real-time operation

They are essential in automation and industrial control.

10. Compare 8255, 8254 and 8257.

Feature	**8255**	**8254**	**8257**
Function	I/O	Timer	DMA
Channels	3 Ports	3 Counters	4 DMA Channels
Purpose	Device Interface	Timing	Fast Data Transfer

SHORT QUESTIONS

1. Total I/O lines in 8255? → 24
2. Counters in 8254? → Three
3. DMA channels in 8257? → Four
4. Mode 0 of 8255? → Basic I/O
5. Square wave mode of 8254? → Mode 3
6. Common types of 7- segment? → Anode & Cathode
7. Device used for motor driving? → ULN2003
8. Stepper motor type? → Unipolar/Bipolar
9. Purpose of DMA? → Direct memory transfer
10. Handshaking used in? → Mode 1